International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 1, ISSN 2249 –071X

A Novel Approach for Solving Travelling Salesman Problem and Compare Its Performance with Inver-Over Operator of Genetic Algorithm

Syed Tauhid Zuhori Department of
Computer Science & Engineering Rajshahi
University of Engineering & Technology

***Abstract* — The traveling salesman problem (TSP) is one of the most widely studied NP hard combinatorial optimization problems and has already solved in the semi-optimal manners using numbers of different methods. Among them, Genetic Algorithms (GA) are pre-dominating. In this paper I solve the problem with a new operator, Inver-over, for an evolutionary algorithm for the TSP. This operator outperforms all other 'genetic' operators, whether unary or binary, which was first introduced by Guo Tao and Zbigniew Michalewicz. I also propose a new algorithm for solving TSP. To get a comparative idea of the performance of these algorithms I solve same problems with the two algorithms. The performance analysis shows that my proposed algorithm produces relatively better solutions in the case of the tour length every time. But when we increase the cities it takes more time to solve than the Inver-Over operator for TSP.**

***Key Words* — Inver-Over operator, Genetic Algorithm, Travelling Salesman Problem.**

I. INTRODUCTION

Traveling salesman problem is not only a popular but also an old problem. Many methods have been used to solve this problem. All methods are included to improve the performance of the previously solved problems. Many Optimal solutions are found. Some solutions are successful to find out the shortest tours and some for finding the solution in the small time. We can describe the history of the traveling salesman problem [1] how can it comes, from the following table—

Year	What was happened
1759	TSP was documented by Euler (not by that name), whose interest was in solving the Knight's tour problem.
1932	The term "Traveling salesman "was first used in a German book written by a veteran traveling salesman.
1948	It was introduced by RAND Corporation.

Table 1.1: History of TSP

The reputation of the RAND corporation helped to make the TSP as a well-known and the popular problem. Day by day the Traveling salesman problem has occupied the thoughts of numerous researches. There are several reasons for this [1, 2, 3]:-

1. TSP is very easy to describe, yet very difficult to solve. There is no polynomial time algorithm is known with which it can be solved. This lack of any polynomial time algorithm is a characteristic of the lass of NP-complete problems, of which the tspis a classic example.
2. The TSP is broadly applicable to a variety of routing and scheduling problems.
3. Since a lot of information is already known about the TSP, it has become a kind of "test" problem; new combinational optimization methods are often applied to the TSP so that an idea can be formed of their usefulness.

At the last 30 years several attempts have been made to develop global optimization algorithms. These optimizations algorithms can simulate the natural optimization process. So the result of these attempts is in the optimization methods [4]:

→ Simulated Annealing, based on natural annealing processes.

→ Artificial Neural Networks, based on processes in central nervous system.

→ Evolutionary Computation based on biological evolution processes.

The algorithm inspired by Evolutionary Computation is called evolutionary algorithms [1]. These evolutionary algorithms may be divided into the following branches:-

- Genetic Algorithm
- Evolutionary Programming
- Evolution Strategies
- Classier Systems
- Genetic Programming
- Other Optimization algorithms based on Darwin's evolution theory of natural selection and survival of the fittest.

II. TRAVELING SALESMAN PROBLEM

The traveling salesman problem (TSP) asks for the shortest route to visit a collection of cities and return to the starting point. Suppose we have a collection of cities and the cost of travel between each pair of them, the traveling salesman problem for short, is to find the cheapest way of visiting all of the cities and returning to your starting point [5].

If we want to express the Traveling Salesman Problem, ‘‘TSP,’’ mathematically then suppose we are given a set {c 1 ,c 2 , . . . ,cN} of cities and for each pair {c i ,c j} of distinct cities a distance d(c i ,c j). Our goal is to find an ordering л of the cities that minimizes the quantity

i = 1

∑ d(c p(i) ,c p(i + 1)) + d(c p(N) ,c p(1)) .

N - 1

This quantity is referred to as the tour length, since it is the length of the tour a salesman

Would make when visiting the cities in the order specified by the permutation, returning at the end to the initial city [6].

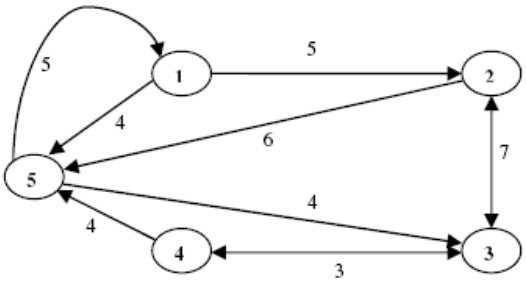


Figure 2.1: Graph for traveling salesman problem [7]

As an example let, G is a directed graph with n vertices. Let length (u, v) be the length of the edge (u, v).A path starting at a given vertex v_0 going throw every other vertex exactly once and finally returning to v_0 is called a tour. The length of a tour is the sum of the length of the edges on the path defining the tour we are concerned with finding a tour of minimum length [8]. In the figure 2.1 we see a city map where 5 cities are included and there costs are given. Here our work is to find out the shortest path as well as shortest cost needed to visit all cities and again return in the starting node. So in a brief we can discuss it as follows [9]:

→ A salesman is assigned N cities to visit.

→ Visit each of the cities once in a continuous trip and return to the origin using shortest

path.

→ DEPOT - starting point for a TSP

→ STOPS - points to be visited

→TOUR - resulting path

→Solution

* Tour detail

* Tour cost

III. GENETIC ALGORITHM

Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. It is a rapidly growing area of artificial intelligence. Genetic algorithms belong to lot of best characteristics that decides it’s a good option when someone needs to solve very complicated problems or NP hard problems. The simplicity and robustness of the algorithm has made it popular among developers. [7]

A basic flowchart of a genetic algorithm has been given here

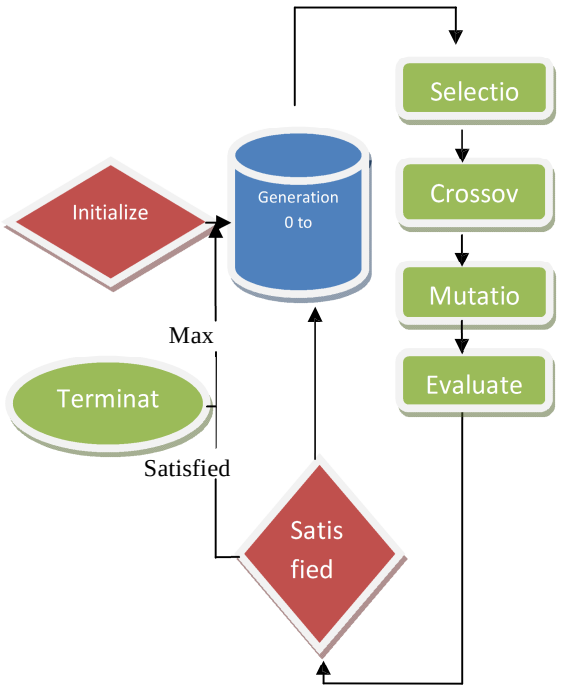


Figure 3.1: A basic flowchart of a Genetic Algorithm. [10]

IV. INVER-OVER OPERATOR

Inver-Over Operator incorporates the knowledge taken from other individualsin the population. One can view this operator as a mixture of inversionand recombination: on one hand, the inversion is applied to a part of a singleindividual; however, the selection of a segment to be

inverted depends on otherindividuals in the population [11].It seems that the proposed algorithm still can't compete (at least as far as computational time is concerned) with efficient approaches based on local search [12]; however, it has a few advantages. First of all, it is extremely simple and easy to implement (less than 100 lines of C code). Additionally, experimental results indicate that this operator outperforms all other evolutionary operators (whether unary or binary), which have been proposed in the past for the TSP (PMX, OX, CX, ER, Edge-2, Edge-3, MPX, RAR, GNX, 2-repair, simple inversion, swap, remove and reinsert, and many others). Moreover, the evolutionary algorithm based on the proposed operator is quite fast (in comparison with other evolutionary techniques) and the quality of results is very high [11].

For described the algorithm Let's illustrate a single iteration of this operator on the following example. Assume that the current individual S' is

S' = {2, 3, 9, 4, 1, 5, 8, 6, 7}

And the current city c is 3. If the generated random number *rand ()* does not exceed *p*, another city c ' from the same individual **S'** is selected (say, c t is 8), and appropriate segment is inverted, producing the following offspring

S' ← {2, 3, 8, 5, 1, 4, 9, 6, 7}

(note the position of the cutting points for the selected segment, which are after cities 3 and 8). Otherwise (i.e., *rand () > p),* another individual is (randomly) selected from the population; assume, it is (1, 6, 4, 3, 5, 7, 9, 2, 8). This individual is searched for the city c' "next" to city 3 (which is 5), thus the segment for inversion in **S'**starts after city 3 and terminates after city 5; consequently, the new offspring is

S' ← {2, 3, 5, 1, 4, 9, 8, 6, 7}

Note again, that a substring 3 - 5 arrived from the "second parent". Note also, that in either case the resulting string is intermediate in the sense that the above inversion operator is applied several times before an offspring is evaluated. This process terminates when the next city c' (to the current city c) in randomly selected individual is also "next city" in the original individual. For example, assume that after a few inversions, the current individual S' is

S' ← {9, 3, 6, 8, 5, 1, 4, 2, 7}

And the current city c is 6. If *rand () > p,* a city 'next' to city 6 is recovered from a (randomly) selected individual from the population; assume, it is city 8 (if *rand ()* < p, a random city is selected, so it may also happen that city 8 was chosen). Since city 8 already follow city 3, the sequence of inversions terminates.

V. PROPOSED ALGORITHM

Here are the steps of our proposed Algorithm.

1. Set i=0;

   Select the starting node and set is as visited

   Visited node[i]=starting node;

   Wining node= visited[i];

   Marked wining node as visited.

   i=i+1;
2. Checking that the all nodes are visited or not
   a. If all nodes are visited then go to step 6
   b. If all nodes are not visited then go to step 3
3. Find the minimum cost from the wining node to the nodes which are not visited
4. If there are more than one node are minimum then set all of them as partially visited and set them in an array. Then for all elements of the array we go to step 2 and do the operation.
   If there is only one node is minimum then go to step 5.
5. Visited node[i]=minimum cost's node;
   Wining node= visited node [i];
   Marked wining node as visited.
   i=i+1;
   Go to step 2;
5. Print the array of visited node [] which is the resulting path.
6. Exit.

VI. EXPERIMENTAL RESULT

We solved the TSP using both the Inver-Over Operator of TSP and Our proposed algorithm for different number of cities ranging from 100 to 1000.

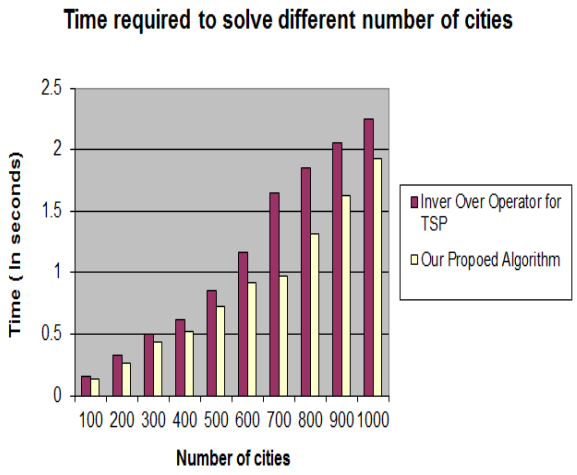
The results are presented in the following table:

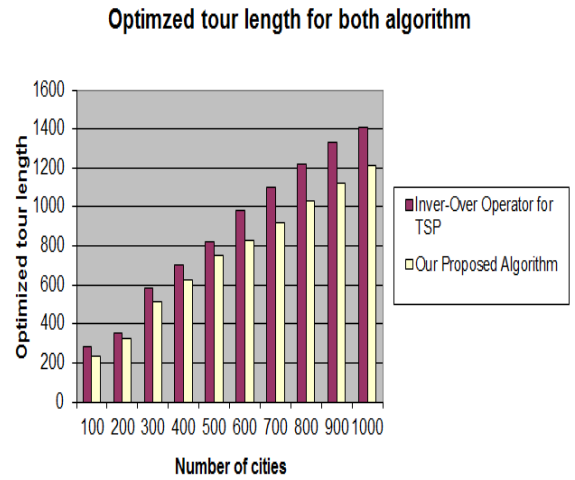
Number Of Cities	Inver-Over Operator in Optimized tour length	Our proposed algorithm in Optimized tour length	Time(In sec) to solve in Inver-Over Operator	Time(In sec) to solve in our proposed algorithm
100	281	232	0.157	0.132
200	352	323	0.322	0.258
300	581	513	0.498	0.437
400	702	622	0.622	0.523
500	823	751	0.858	0.723
600	982	831	1.165	0.922
700	1102	921	1.643	0.975
800	1222	1031	1.852	1.312

900	1332	1121	2.054	1.625
1000	1405	1213	2.243	1.925

Table 6.1: Comparison between Inver-Over Operator of TSP and Our Proposed Algorithm

These values are obtained by taking the average value for several trials for each of the 10 to 100 cities problem. The graphs are designed by us. The time in second was also measured to solve each of the 10 to 100 cities problems. These values are plotted in the following graphs to determine the comparison between the two algorithms.





CONCLUSION

The Inver-Over operator is a special operator for TSP but it gives quick solutions than the others Genetic Operators. In this thesis, we work with this operator and also propose an algorithm. For implementing the algorithm we use the visual c++ and also use file to give inputs. So it is really time consuming to find the minimum values. Because every time we have to use arrays and after finding minimum we also search them from array. If we save our tour using database then we can solve the problems of finding minimum values and search them from list because then we can index them easily and sort them easily. So now I am trying to use database to reduce time complexity. We were able to solve TSPO with 100 cities and with our graph. But now we are trying to solve TSP with formal graph which WAS solved by the researchers. We are also trying to solve TSP with more than 100 cities.

REFERENCES

[1] Abdullah Al Mohammad, "Performance Comparison of Genetic Algorithm and Simplified Bi-directional Associative Memory (sBAM) in solving Traveling Salesman Problem". Page:2-3
[2] Dominique Feillet, Pierre Dejax and Michel Gendreau, "Traveling Salesman Problems with Profits", page 1.
[3] David S. Johnson and Lyle A. McGeoch2, "The Traveling Salesman Problem: A Case Study in Local Optimization", page 1.
[4] James A. Freeman and Davis M Skapura, " Neural Networks, Algorithms, Applications and Programming Techniques", page-2
[5] http://www.tsp.gatech.edu/problem/index.html
[6] David S. Johnson and Lyle A. McGeoch, "The Traveling Salesman Problem: A Case Study in Local Optimization", Page-3.
[7] Friedel Wolf, "The Column Subtraction Method for the Traveling Salesman Problem", Page-2
[8] FUNDAMENTALL OF COMPUTER ALGORITHMS by SartajSahni,EllisHorowitz,SanguthevarRajasekaran
[9] Dr. Tom V. Mathew, "Genetic Algorithms".
[10] Applegate, D., Bixby, R., Chvatal, V., and Cook, W. Concorde tsp solver.
[11] Guo Tao and Zbigniew Michalewicz, "Inver-over Operator for the TSP".
[12] Johnson, D.S., The Traveling Salesman Problem: A Case Study. In *Local Search in Combinatorial Optimization*, E. Aarts and J.K. Lenstra (Editors), John Wiley,1996, pp.215-310.

AUTHOR'S PROFILE

Syed Tauhid Zuhori
Department of Computer science & Engineering
Rajshahi University of Engineering & Technology (RUET),
Bangladesh.
tauhid.ruet@yahoo.com